**Fisa 8**

**Notiuni teoretice**

Functii care lucreaza cu siruri de caractere (pentru acestea se include biblioteca <cstring>):

1. **strlen**

size\_t [**strlen**](http://www.opengroup.org/onlinepubs/009695399/functions/strlen.html)(const char \*str);

Returneaza lungimea unui şir dat ca parametru. (numarul de caractere până la întalnirea terminatorului de şir:\0)

size\_t il puteti privi ca un unsigned int

***Ex.***

char s[50]="Un exemplu";

int n=strlen(s);

cout << "Sirul s are "<<n<<" caractere " << endl;

1. **strcpy**

char\*[**strcpy**](http://www.opengroup.org/onlinepubs/009695399/functions/strcpy.html)**(** char \*destination**,** const char \*source**);**

Copiază şirul sursă in şirul destinaţie, incepand cu pozitia 0. Funcţia asigură plasarea terminatorului de şir în şirul destinaţie după copiere. Şirul destinaţie va fi suprascris. Funcţia returneaza şirul destinaţie.

***Ex. 1***

char d[50]="Un sir", s[50]="Alt sir de caractere";

strcpy(d,s);

cout<<d; // se va afisa: Alt sir de caractere

***Ex. 2***

char d[50]="Un sir", s[50]="Alt sir de caractere";

strcpy(s,d);

cout<<d; // se va afisa: Un sir

1. **strncpy**

char\*[**strncpy**](http://www.opengroup.org/onlinepubs/009695399/functions/strcpy.html)**(** char \*destination**,** const char \*source, size\_t n**);**

Asemeni cu **strcpy**(), dar in loc de a fi copiată toata sursa sunt copiate doar primele ***n*** caractere.

***Ex.***

char d[50]="mere", s[50]="banana";

strncpy(d,s,2);

cout<<d<<endl; // se va afisa: bare

d[2]='\0' // se va afisa: ba

strcpy(s,”nuci”);

cout<<s; // se va afisa: nuci

1. **strcat**

char\* **strcat(** char \*destination**,** const char \*source**);**

Concatenenaza şirul sursă la şirul destinaţie. Funcţia returnează şirul destinaţie.

***Ex.***

char a[50]="Ana", b[50]="Maria";

strcat(a,b);

cout<<a<<endl; // se va afisa: AnaMaria

1. **strncat**

char\* **strncat(** char \*destination**,** const char \*source, size\_t n **);**

Asemeni cu strcat(), dar în loc de a fi concatenată toată sursa sunt concatenate doar primele n caractere.

***Ex.***

char a[50]="Ana are", c[50]=" si pere si struguri ";

strcat(a," mere");

cout<<a<<endl; // se va afisa: Ana are mere

strncat(a, c, 8);

cout<<a; // se va afisa: Ana are mere si pere

1. **strchr**

char\* **strchr(** const char \*s, int caracter**);**

Caută caracterul în şirul s şi returnează un pointer la *prima* sa apariţie.

***Ex.***

*char a[]="Aceasta este";*

*char \*p;*

*p=strchr(a,'t');*

*cout<<p<<endl; // se va afisa: ta este*

*// se poate calcula indicele primei aparitii a caracterului in sir*

*cout<<"indice "<<p-a; // se va afisa 5 (deoarece numerotarea incepe de la 0)*

1. **strrchr**

char\* **strrchr(** const char \*s, int caracter**);**

Caută caracterul *character* în şirul *s* şi returnează un pointer la *ultima* sa apariţie.

1. **strstr**

char\* **strstr(** const char \*s1, const char \*s2**);**

Caută şirul s2 în şirul s1 şi returnează un pointer la prima sa apariţie, sau *NULL* dacă nu a fost găsit.

1. **strcmp**

int\* **strcmp(** const char \*s1, const char \*s2**);**

Compară şirul s1 cu şirul s2, verificându-le caracter cu caracter. Valoarea returnată este 0 daca cele şiruri sunt identice, mai mare ca 0 daca str1 este “mai mare”(alfabetic) şi mai mic ca 0 altfel.

***Atentie!*** Se face distinctie intre litere mari si litere mici, iar “Ana” este considerat ca fiind inaintea cuvantului “ana” (deci “ Ana<ana” deoarece literele mari au un cod ASCII mai mic decat literele mici).

***Ex.***

char s1[50], s2[50];

int r;

cin.getline(s1,50);

cin.getline(s2,50);

r=strcmp(s1,s2);

if (r>0)

cout<<" s1>s2 - in sens lexicografic";

else

if (r==0)

cout<<" s1 si s2 - sunt identice";

else

cout<<" s1<s2 - in sens lexicografic";

1. **stricmp**

int\* **stricmp(** const char \*s1, const char \*s2**);**

La fel ca **strcmp,** dar nu se face distinctive intre litere mari si mici si de aceea daca folosim aceasta functie pentru a compara “Ana” cu “ana” ne va intoarce valoarea 0, adica le considera identice.

1. **strrev**

char\* **strrev(** char \*s**);**

Intorce inversul sirului s'.

***Ex.***

char s1[50], \*p, s2[50];

cin.getline(s1,50); //daca se citeste ABC

strcpy(s2,s1);

p=strrev(s1);

cout<<p; // se afiseaza CBA

cout<<endl<<s1; // se afiseaza CBA

if(strcmp(s1,s2)==0)

cout<<"cuvant <<palindrom>>";

1. **strlwr**

char**\* strlwr (** char**\*s );**

  Converteste literele majuscule ale sirului s la litere mici. Returneaza adresa sirului s.

1. **strupr**

char**\* strupr (** char**\*s );**

  Converteste literele minuscule ale sirului s la litere majuscule. Returneaza adresa sirului s.

1. **strset**

char **\* strset** **(** char **\*s,** int **ch );**

Seteaza toate caracterele sirului s la caracterul ch si intoarce ca rezultat adresa lui s.

1. **strnset**

char **\* strset** **(** char **\*s,** int **ch ,** size\_t n **);**

La fel ca strset doar ca seteaza exact primii n octeti ai sirului s la caracterul ch. Daca n > strlen (s) operatia se opreste la sfirsitul sirului.